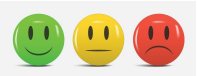
<u>Introduction to Deep Learning - Exercise #2</u>

submission date: 10/12/2021

Programing Task: Sentiment Analysis for the IMDB Movies Reviews Dataset



The dataset consists of 50,000 annotated reviews. Each review is a plain text describing the viewers' experience and opinion on the movie watched. The dataset consists of highly polar reviews and contains binary labels (positive and negative) based on the number of stars the movie received by the viewers. Some reviews are very long, but we will consider only the first 100 words they contain.

In this exercise we design networks that predict the viewers' sentiment (positive/negative) towards the movies they watched based on the review they wrote. We will compare the following four different strategies:

1. The use of a simple Elman RNN
2. The use of a simple GRU
3. The use of a global average pooling. In this case every word goes through an MLP that result in a scalar, *sub-prediction score*, and these scores are then summed together to provide the final prediction
4. Adding a local self-attention layer to Strategy #3 in order to achieve crossword reasoning

The exercise comes with a partial code that loads the reviews from the dataset, and processes them into a long list of lower-case text (100 words, no punctuation/sentences, no special characters). Longer reviews are truncated, shorter ones are padded. This pre-processing uses the GloVe word embedding which maps every word into a 100-dimensional vector. To avoid over-fitting over this small dataset we do not allow this embedding to train. You can read more about this embedding here. Some parts of these pre-processing steps were taken from this tutorial.

Specific tasks:

1. Fill in the missing lines of code in the RNN and GRU cells functions. The RNN contains some lines which you may find helpful (or choose to omit and implement on your own). The gates and update operators should consist of a single FC layer (hidden state dim. should be between 64-128 for the lowest test error). The convention of the tensors for these recurrent networks is: **batch element x "time" x feature vector**. So the recurrence (your iteration in the code) should apply on the second axis. Once the review is parsed, its hidden-state should pass through an MLP (with a sigmoid activation as its end) which produces the final output sentiment prediction (a scalar inside [0,1]).
   Run each of these two recurrent network architectures,describe your experiments with the hidden-state dimension and the train/test accuracies obtained.

2. In a second experiment you will process each word by an MLP to obtain a scalar value, "*sub prediction score*", and then sum up all these values to obtain a final prediction (which still allows us to handle data of variable length). A sigmoid activation should be applied after the summation to limit the output prediction into [0,1] (subscores can be either positive or negative values).
   Describe your experiments with the number of FCs layers and their inner dim and report the best performing one you found.
   Since this model gives a score per each word, output two test examples with these numbers next to their corresponding words (`reviews_text` and `sub_score` in the code). The examples should be selected such that one which the prediction is right and one which is wrong. Come up with an explanation of why this happened. To conduct this experiment you are welcome to write your own test reviews in loader.py in the `my_test_texts` list.

3. Write a restricted self-attention layer which queries every word with its closest 5 words on each side (using torch.roll and padding of size 5). This layer should have a single head and learnable query and key matrices. You can use the incomplete code in `ExLRestSelfAtten` to implement this layer or use your own code. This code can use torch.roll to shift the text right and left as well as torch.pad to handle the boundaries.

4. Finally, add this layer to the network architecture you used in Task 3 and repeat the experiments above; two reviews, one correctly predicted, one wrong, print the sub prediction scores per each word, and explain the results - how they differ from before. Include both these printouts in your report as well as describe the main principle difference in the predictions abilities that this layer adds to the network and how it can be seen in the results.

We are expecting you to report and elaborate on every practical task in the pdf, with your own words and analysis of what you've done. Include everything that you think is crucial for us to understand your way of thinking.

Theoretical Questions:

1. Explain what type of a network architecture you will use to handle each of the following problems (e.g., many-to-many RNN, or a convolution NN). Explain your reasoning.
   a. Speech recognition (audio to text)
   b. Answer questions
   c. Sentiment analysis
   d. Image classification
   e. Single word translation
2. Compute the analytical Jacobian of the following layers:
   a. Linear layers Ax
   b. Bias layers x+b
   c. Convolutional layers x * f

3. Text-to-Image:
    a. Describe the architecture of a network that reads a sentence and generates an image based on the text. Do not address the question of how such a network is trained, just explain why it should have the capacity to perform this task. You can assume that the images come from a restricted class of images, e.g., faces, and can be encoded (and decoded) in a low-dimensional latent space.

    b. Assume the image is encoded using 4 latent codes that correspond to its four quadrants. Explain how an attention layer can be used to allow such a network to better support fine-grained descriptions in the input text, as well as references to different regions (top, bottom, left, right, sky, ground, etc.). What would be the queries, keys, and values in this case?
4. CNNs:
    a. Assume an 128X128x1 input image is inputted to the following architecture :

        conv(kernal_size=3)
        conv(kernal_size=3)
        maxpool(sampling_rate=2)
        conv(kernal_size=3)
        conv(kernal_size=5)

        where each of the conv operators works in “valid” mode, i.e., it computes an output response if all the pixels in its receptive field exist in its input. What would be the spatial resolution of this network.

    b. What would be the size of the receptive field of each neuron in the final layer (in terms of pixels in the input image)?

    c. Describe a method to estimate the importance (contribution) of each region in the input image to the final prediction of the network (on that image).

Submission Guidelines:

The submission is in **<u>pairs</u>.** Please submit a single zip file named “ex2_ID1_ID2.tar”. This file should contain your code, along with an ”ex2.pdf” file which should contain your answers to the theoretical part as well as the figures/text for the practical part. Furthermore, include in this compress file a README with your names and cse usernames.
Please write readable code, with documentation where needed, as the code will also be checked manually.